INTRODUCCIÓN AL ANÁLISIS NUMÉRICO

F. VADILLO

Resumen. En esta introducción al Análisis Numérico primero se cometa el objetivo de la esta disciplina y después se *esbozan* algunos conceptos muy generales y se introducen herramientas matemáticas. Se comienza comentando la idea de problemas mal o bien planteados, después se trata del concepto de la convergencia y finalmente se describen las fuentes de error con una especial atención a los errores de redondeo inevitables en el sistema numérico de coma flotante. En la última parte se recuerdan los concepto de normas vectoriales y se introducen las normas matriciales que se utilizan en capítulos posteriores.

Índice

1. Sobre el Análisis Numérico

El objetivo del Análisis Numérico, el Cálculo Numérico, los Métodos Numéricos o las Ciencias de la Computación, es fácil de establecer, se trata de utilizar el computador para resolver problemas matemáticos [4]. El profesor Nick Trefethen, último presidente del SIAM (Society for Industrial and Applied Mathematics), en un apéndice de su libro [8] propone varias definiciones de Análisis Numérico y después de algunos disquisiciones se decide por la siguiente: El Análisis Numérico es el estudio de algoritmos para resolver problemas de las matemáticas continuas.

Una definición más detallada de lo que se entiende por Cálculo o Análisis Numérico podría decir que: El Análisis Numérico es la rama de las Matemáticas que

Received by the editors 13 de septiembre de 2022.
www.ehu.es/~mepvaarf.

desarrolla y analiza algoritmos y métodos para obtener soluciones numéricas de problemas matemáticos, ¿por qué soluciones numéricas? evidentemente porque sólo en muy pocos modelos matemáticos las soluciones se pueden expresar mediante funciones elementales, es decir usando polinomios, funciones trigonométricas, logarítmicas, exponenciales...

El profesor Jesus Sanz-Serna en el prólogo de [7] escribe que aprender matemáticas es un proceso donde se trata de aprender a hacer algo, no sólo de adquirir conocimientos. Por eso destaca que: El Calculo Numérico además de una disciplina académica que se estudia en las aulas, es parte de un esfuerzo humano global desarrollado a lo largo de la historia para resolver problemas científicos y técnicos y sobretodo destaca que las teorías se han construido en la historia para resolver problemas específicos: puede haber Matemáticas sin teoremas más no sin problemas.

El Análisis Numérico es una disciplina Matemática ubica en el área de la Matemática Aplicada que resuelve una gran variadas de problemas matemáticos

- Sistemas de ecuaciones lineales.
- Ecuaciones y sistemas no lineales.
- Problemas de ajuste de curvas a datos experimentales.
- Aproximación de funciones.
- Problemas de optimización.
- Ecuaciones diferenciales.
- .....

Aunque los métodos numéricos son diferentes según el problema que se resuelve, se puede hablar de algunos conceptos generales en todo el Análisis Numérico que se comentan a continuación, las definiciones exactas se deberán encontrar en cada caso concreto.

2. Condicionamiento de un problema

De manera genérica se considera el problema siguiente: conocidos unos datos $\mathbf{d}$ se pide calcular los valores de $\mathbf{x}$ tal que

$$\mathbf{F}(\mathbf{x}, \mathbf{d}) = \mathbf{0}, \tag{2.1}$$

donde $\mathbf{F}$ es una función que determina el modelo matemático y relaciona los datos $\mathbf{d}$ con los resultados $\mathbf{x}$. Evidentemente, las variables $\mathbf{d}$ y $\mathbf{x}$ pueden ser de distintos tipos y dimensiones.

Se dice que el problema (2.1) está **bien condicionado** o **bien puesto**, si la solución $\mathbf{x}$ es única para cada dato $\mathbf{d}$ y además pequeños errores en las mediciones de $\mathbf{d}$ provocan cambios moderados en el resultado $\mathbf{x}$. Para simplificar la exposición, en los comentarios posteriores las variables x y d se consideran escalares, para extender los conceptos a problemas vectoriales prácticamente basta sustituir los valores absolutos $|\cdot|$ por normas vectoriales $\|\cdot\|$ y poco más.

Ejemplo 2.1. El polinomio $p(x) = x^4 - x^2(2a-1) + a(a-1)$ tiene las raíces $x = \pm\sqrt{a}$ y $x = \pm\sqrt{a-1}$. El número de raíces reales diferentes depende del valor de a, si $a > 1$ tiene cuatro raíces reales distintas, si $0 \leq a < 1$ tiene dos y no existen raíces reales si $a < 0$, por tanto el problema de determinar el número de raíces reales de $p(x)$ en función de a es un problema mal puesto porque para a próximo a cero un pequeño error en la medición hace saltar de cero a dos en el número de soluciones reales.

Para cuantificar la *calidad* de un problema se considera una perturbación pequeña del dato δd, ahora la solución es $x+\delta x$, es decir, $F(x+\delta x, d+\delta d)=0$ y se define el **número de condición relativo** de la forma siguiente

$$K(d)=\sup_{\delta d\in D}\frac{|\delta x|/|x|}{|\delta d|/|d|}, \tag{2.2}$$

donde D es un entorno del origen que indica el conjunto de las perturbaciones. Si $K(d)$ es un cantidad elevada se dice que el problema está mal puesto o mal condicionado lo que hace imposible su tratamiento numérico. Por contra, cuando el número de condición sea *moderado* el problema tiene un posible tratamiento numéricamente.

Si el problema (2.1) tiene una única solución x para cada dato o conjunto de datos d, necesariamente existe una función G llamada **resolvente** tal que

$$x=G(d) \qquad \Rightarrow \qquad F(G(d),d)=0. \tag{2.3}$$

Suponiendo ahora que G es diferenciable, utilizando el teorema del valor medio se tiene que

$$\delta x=(x+\delta x)-x=G(d+\delta d)-G(d)\approx G'(d)\delta d \tag{2.4}$$

de donde se deduce que el número de condición relativo se puede aproximar de la siguiente forma

$$K(d)\approx|\,G'(d)\,|\,\frac{|\,d\,|}{|\,G(d)\,|}. \tag{2.5}$$

Ejemplo 2.2. Las soluciones de la ecuación algebraica $F(p,x)=x^2-2px+1=0$ con $p>1$ son $x_\pm=p\pm\sqrt{p^2-1}$, es decir, la resolvente de esta ecuación es

$$G_\pm(p)=p\pm\sqrt{p^2-1} \quad \Rightarrow \quad G'_\pm(p)=1\pm\frac{p}{\sqrt{p^2-1}},$$

y la aproximación del número de condición resulta

$$K(p)\approx\frac{|\,p\,|}{\sqrt{p^2-1}}.$$

Cuando $p\approx 1$ el denominador $\sqrt{p^2-1}\approx 0$ y $K(p)$ es muy grande por lo cual el problema está mal condicionado, pequeños errores en el parámetro p provocan cambios enormes en el resultado.

Introduciendo el nuevo parámetro $t=p+\sqrt{p^2-1}$ la ecuación que resulta es $F(t,x)=x^2-\frac{1+t^2}{t}x+1=0$ que tiene las raíces $\{x_-=t, x_+=\frac{1}{t}\}$. La estimación del nuevo número de condición ahora es $K(t)\approx 1$ y por tanto el problema transformado está bien puesto. Aunque matemáticamente el problema es el mismo numéricamente son problemas muy diferentes.

Para resolver numéricamente un problema mal condicionado es imprescindible transformarlo. Es un error pensar que un método numérico pueda curar la patología de un problema mal puesto.

3. Estabilidad de los métodos numéricos

Suponiendo que el problema escalar (2.1) está bien puesto, en general, los métodos numéricos calculan soluciones aproximadas que resuelven una sucesión de problemas *parecidos* de alguna manera al problema original

$$F_n(x_n, d_n) = 0 \qquad \text{para} \qquad n \geq 1, \tag{3.1}$$

donde el parámetro n difiere en cada situación. Ahora lo que se espera es que las soluciones numéricas x_n se vayan acercando a la solución exacta x, es decir, que haya una convergencia en algún sentido.

Para estudiar la convergencia se definen los errores absolutos

$$E(x_n) = \mid x - x_n \mid, \tag{3.2}$$

así como los errores relativos que son

$$E_{rel}(x_n) = \frac{\mid x - x_n \mid}{\mid x \mid} \qquad \text{si} \qquad x \neq 0. \tag{3.3}$$

La ventaja de los errores relativos es que son independientes de la escala, si $x \mapsto \alpha x$ y $x_n \mapsto \alpha x_n$ no cambia, mientras que los errores absolutos se multiplican por el factor de escala α.

El error relativo está relacionado con el concepto de dígitos significativos correctos de una aproximación, expresión muy utilizada pero muy poco precisa. Se entiende por dígitos significativos de un número a sus primeros dígitos no cero, por ejemplo 1.7320 tiene cinco y 0.0491 tiene sólo tres. Si se quisieran comparar los dígitos de x y su aproximación x_n lo cosa tiene su misterio que puede llevar a situaciones confusas. Vea los dos casos siguientes

$$\begin{array}{lllll} (1) & x = 1.0000 & x_n = 1.00499 & E_{rel} = 4.99 \times 10^{-3} \\ (2) & x = 9.0000 & x_n = 8.99899 & E_{rel} = 1.12 \times 10^{-4} \end{array}$$

En el primer caso x y su aproximación tiene iguales los tres primeros dígitos mientras que el segundo no tiene ninguno a pesar de tener un error relativo menor. Parece razonable pensar que algo se debe cambiar en el concepto de dígitos significativos de una aproximación, una opción sería pensar que una aproximación tienen p dígitos correctos si al redondear a p dígitos el número y su aproximación coinciden. Redondear significa tomar el número de p dígitos más proximo.

Esta nueva versión resuelve el ejemplo anterior porque en el primer ejemplo redondeado a tres dígitos da $x = x_n = 1.00$ y en el segundo redondeado a cuatro da $x = x_n = 9.000$. Sin embargo, tomado ahora $x = 0.9949$ y $x_n = 0.9951$ resultaría que la aproximación tiene tres dígitos correctos pero no dos.

Pueden intentar otras definiciones aunque siempre se la puede encontrar el contraejemplo que la deje maltrecha. En conclusión, aunque en términos coloquiales se hable del número de dígitos correctos de una aproximación, siempre resulta poco preciso y es preferible hablar de errores.

Para estudiar los errores $e_n = x - x_n$, o bien se conoce la solución exacta x, en cuyo caso poco puede interesar buscar soluciones aproximadas, o quizá se puedan acotar los errores por cantidades que van a cero cuando $n \to \infty$, si bien este segundo camino, en general, resulta muy complicado y poco práctico. Frecuentemente la convergencia de un método numérico se demuestra a través de dos nuevos conceptos que son la consistencia y la estabilidad que se comentan a continuación.

El método (3.1) se dice que es consistente con el problema (2.1) si

$$F_n(x,d) \longrightarrow 0 \qquad \text{cuando} \qquad n \to \infty, \tag{3.4}$$

es decir, se comprueba en qué medida los datos y la solución exacta verifican la ecuación aproximada (3.1), si dicho error tiende a cero cuando $n \to \infty$ se tiene la consistencia. Si se hiciera al revés, es decir, se calculara $F(x_n, d_n) \neq 0$, se mediría la exactitud con la que las soluciones aproximadas verifican la ecuación exacta (2.1) y se obtendrían los residuos que es otra forma de calibrar la *calidad* de una aproximación.

En algunos caso, por ejemplo en los métodos iterativos, la ecuación (3.1) que define el método numérico puede tener otra forma

$$F_n(x_n, x_{n-1}, \ldots, x_{n-q}, d_n) = 0 \tag{3.5}$$

donde $x_0, x_1, \ldots, x_{q-1}$ son datos. En este caso la consistencia calcularía $F_n(x, x, \ldots, x, d)$ para $n \geq q$.

Ejemplo 3.1. El **método de Newton** es muy habitual para aproximar la raíz simple α una ecuación $f(x) = 0$. El método calcula una sucesión de aproximaciones con la iteración

$$\begin{cases} x_0 \quad \text{dado}, \\ x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}, \qquad n \geq 0, \end{cases} \tag{3.6}$$

Ahora $F_n(x_n, x_{n-1}, f) = x_n - x_{n-1} + \frac{f(x_{n-1})}{f'(x_{n-1})}$, entonces $F(\alpha, \alpha, f) = 0$ y el método es siempre consistente.

Por otra parte, para que el método (3.1) sea práctico se necesita que para cada dato o conjunto de datos d_n exista una única solución x_n, es decir, debe existir otra resolvente G_n del método tal que

$$x_n = G_n(d_n), \tag{3.7}$$

de forma que $F_n(G_n(d_n), d_n) = 0$ para todo n. Además, las soluciones aproximadas x_n deberán cambiar poco cuando los datos d_n sean ligeramente modificados, los métodos numéricos con este tipo de comportamiento se les llama métodos estables. Dicho en otras palabras, es imprescindible que los métodos numéricos sean estables para que no engañen en los resultados.

Para cuantificar la estabilidad de un método se define su número de condición de manera similar a como ya se hizo para los problemas matemáticos

$$K_n(d_n) = \sup_{\delta d_n \in D_n} \frac{|\, \delta x_n \,| \,/\, |\, x_n \,|}{|\, \delta d_n \,| \,/\, |\, d_n \,|}, \tag{3.8}$$

y cuando esta cantidad sea *grande* el método es inestable y de muy escasa utilidad práctica.

Si la resolvente G_n es diferenciable, repitiendo los cálculos anteriores se consigue una estimación del número de condición que es

$$K_n(d_n) \approx |\, G_n'(d_n) \,| \, \frac{|\, d_n \,|}{|\, G_n(d_n) \,|}. \tag{3.9}$$

Ejemplo 3.2. El problema de las cancelaciones. La función $f(a,b) = a+b$ es lineal y su gradiente (derivada) es $(1,1)^T$. En [6] se comprueba que para la norma $\|\cdot\|_1$ el número de condición es

$$K(a,b) \approx \frac{\mid a \mid + \mid b \mid}{\mid a+b \mid}. \tag{3.10}$$

Por tanto, sumar dos números de igual signo es estable porque $K(a,b) \approx 1$, pero al restar dos números casi iguales $\mid a+b \mid \ll \mid a \mid + \mid b \mid$ por lo que el número de condición es muy grande y la operación es inestable, esto significa que cuando se restan dos números muy parecidos los errores se propagan deforma catastrófica y es ello una operación que se debe evitar.

Los conceptos de estabilidad y convergencia están fuertemente relacionados. En primer lugar, siempre que el problema este bien condicionado la estabilidad del método numérico es necesaria para la convergencia, es decir, no existen métodos convergentes que sean inestables. En efecto, suponiendo que el método es convergente, llamando $x_n(d)$ a la solución numérica con el dato d se tiene

$$\begin{aligned} \mid \delta x_n \mid &= \mid x_n(d) - x_n(d+\delta d_n) \mid \\ &\leq \mid x_n(d) - x(d) \mid + \mid x(d) - x(d+\delta d_n) \mid \\ &+ \mid x(d+\delta d_n) - x_n(d+\delta d_n) \mid, \end{aligned}$$

de donde se deduce la estabilidad porque de estos tres sumandos, el primero y tercero son pequeños por la convergencia y el segundo está acotado porque se ha supuesto que el problema está bien condicionado.

Sin embargo el objetivo no este resultado, si no mas bien el contrario, es decir, deducir un resultado de convergencia a través de la estabilidad. El resultado que habitualmente se denomina teorema de convergencia dice que si un método es consistente, la estabilidad es suficiente para la convergencia, dicho que otra manera, consistencia más estabilidad implica convergencia, porque

$$\begin{aligned} \mid E(x_n) \mid &= \mid x(d+\delta d_n) - x_n(d+\delta d_n) \mid \\ &\leq \mid x(d+\delta d_n) - x(d) \mid + \mid x(d) - x_n(d) \mid \\ &+ \mid x_n(d) - x_n(d+\delta d_n) \mid, \end{aligned}$$

y de estos tres sumandos, el primero está acotado porque el problema está bien puesto, el segundo por la consistencia y el tercero por la estabilidad.

Conclusión: si el método numérico (3.1) es consistente con el problema (2.1) y además es estable, entonces es convergente.

4. Fuentes de error en la computación

El método numérico (3.1) calcula soluciones aproximadas del problema matemático (2.1) que a su vez es un modelo de un problema físico; por tanto, las soluciones aproximadas $\hat{x}_n$ están *contaminadas* de varias fuentes de error que se resumen a continuación

Problema físico
(1) ↓ (2)
Modelo matemático
(3) ↓
Método numérico
(4) ↓
Solución computada

donde

- (1) Errores debido al modelo.
- (2) Errores en los datos, que son mejorables con mediciones más precisas.
- (3) Errores de discretización que es el resultado de cambiar la ecuación exacta (2.1) por la aproximada (3.1).
- (4) Errores de redondeo cometidos en la operaciones para resolverla ecuación aproximada (3.1).

Los errores (3) y (4) son los llamados errores de computación, el error (3) lo controla la convergencia del método, un método numérico se dice que es convergente cuando los errores de discretización se hacen tan pequeños como se quiera con una elección adecuada del parámetro de discretización.

Los errores de redondeo son debidos a la computación finita, en un ordenador sólo se pueden usar un número finito de cifras, es por tanto imprescindible efectuar redondeos. El efecto del redondeo sobre los algoritmos numéricos puede ser dramático como se puede apreciar en el ejemplo siguiente

Ejemplo 4.1. En un ordenar tome un número arbitrario $x > 0$ y haga la siguiente sucesión de operaciones:

$$\begin{array}{ll} \text{para} & i = 1:60 \\ & x = \sqrt{x} \\ \text{final} & \\ \text{para} & i = 1:60 \\ & x = x^2 \\ \text{final} & \end{array}$$

esto es, calcule primero 60 raíces cuadradas y después otros 60 potencias cuadradas. Observará con sorpresa que el ordenador no devuelve el valor correcto que es x, el resultado es 0 cuando $0 < x < 1$ y 1 para $1 \leq x$. El motivo fundamental de este comportamiento es que estos cálculos superan la capacidad de casi cualquier máquina, en la página 18 de la referencia [3] puede consultar los detalles.

5. Representación de los números en un ordenador

En un ordenador sólo se pueden representar un número finito de cifras, así pues, los números con infinitas cifras diferentes se deben aproximar por otro con un número finita de cifra lo que provoca los inevitables errores de redondeo, razón por lo que es muy importante conocer su propagación en los algoritmos numéricos.

Sea x un número real con un número finito de dígitos, escrito x en una base β con la habitual notación posicional

$$x = (-1)^s \cdot x_n x_{n-1} \dots x_0 . x_{-1} \dots x_{-m} = (-1)^s \sum_{k=-m}^{n} x_k \beta^k, \tag{5.1}$$

donde $0 \leq x_k < \beta$ para $k = -m, \ldots, n$ y s depende el signo de x, $s = 0$ para los números positivos y $s = 1$ para los negativos.

Aunque en teoría todas las bases son equivalentes en computación se usan tres

1. Binaria ($\beta = 2$). Los dígitos se reducen a los símbolos 0 y 1 llamados **bits** (binary digits).
2. Decimal ($\beta = 10$).
3. Hexadecimal ($\beta = 16$) con los dígitos 0,1,2,..,9,A,B,C,D,E y F.

5.1. Sistema de numeración de coma flotante. Suponiendo que un ordenador tuviera N posiciones de memoria para almacenar cualquier número, lo más natural sería fijar una posición para guardar el signo, $N - k - 1$ posiciones para la parte entera y las k restantes para la parte decimal. Esta representación denominada sistema de punto fijo porque fija el número de dígitos a la izquierda y la derecha de la coma, tiene una fuerte limitación sobre las cantidades que se pueden representar, salvo que N fuera muy grande en cuyo caso se necesitaría una memoria enorme. Para evitar este grave inconveniente se utiliza la representación en punto flotante en donde el número x se escribe en notación exponencial de la siguiente forma

$$x = (-1)^s \cdot (0.a_1 \cdots a_t) \cdot \beta^e = (-1)^s \cdot m \cdot \beta^{e-t}, \tag{5.2}$$

donde t es el número de dígitos significativos, $m = a_1 \cdots a_t$ es la mantisa con $a_1 \neq 0$ y e es el exponente que estará en un intervalo finito $L \leq e \leq U$. El conjunto de números de punto flotante es entonces

$$\mathbb{F}(\beta, t, L, U) = \{0\} \bigcup \left\{ x \epsilon \Re : x = (-1)^s \beta^e \sum_{i=1}^{t} a_i \beta^{-i} \right\}$$

El programa `floatgui.m` de [5] abre una ventana como la representada en la figura 1 donde se representa la distribución de los números positivos en coma flotante para distintos valores de t y diferentes recorridos del exponente.

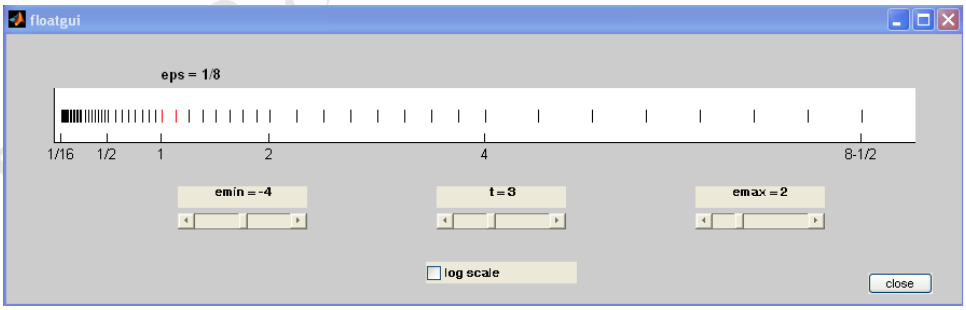

FIGURA 1. Ventana del programa `floatgui`

Cuando se utilizan la representación en punto flotante las N posiciones de memoria se distribuyen de manera diferente: una posición para el signo, t posiciones para la mantisa y las $N - t - 1$ posiciones restantes para el exponente. Las dos formas más utilizadas son las denominadas precisión simple y doble que en caso binario se distribuyen de la siguiente forma

precisión simple:	$N = 32$	1 bit para s	8 bits para e	23 bits para m
precisión doble:	$N = 64$	1 bit para s	11 bits para e	52 bits para m

5.2. Aritmética IEEE. Hace veinte años la situación era muy confusa porque cada computar tenía su propio sistema de punto flotante, había muchos que utilizaban el sistema binario, pero también existían ordenadores que usaban bases 8 y 16, incluso hubo un ordenador ruso en base 3. Para evitar esta confusion en 1985 el IEEE (Institute of Electrical and Electronics Engineers) y la ANSI (American Nacional Standard Institute) adoptaron el convenio ANSI/IEEE Standard 754-1985 para la aritmética binaria en punto flotante que utilizan todos los ordenadores diseñados desde entonces y que es la siguiente

en precisión simple:	$\mathbb{F}(2, 24, -125, 128)$
en precisión doble:	$\mathbb{F}(2, 53, -1021, 1024)$

el $24 = 23 + 1$ y el $53 = 52 + 1$ se deben a que el primer dígito de la mantisa es siempre 1 y no es necesario almacenarlo y para la precisión simple $2^8 = 256$ mientras que para la precisión doble $2^{11} = 2048$. Los rangos de representación son del orden de $10^{\pm 38}$ y $10^{\pm 308}$ respectivamente.

Algunos números especiales como ± 0 , $\pm\infty$ y el NaN (not a number) que representa operaciones no validadas como por ejemplo $0/0, \infty - \infty, 0 \cdot \infty \cdots$ tienen las siguientes representaciones

valor	exponente	mantisa
± 0	L-1	0
$\pm\infty$	U+1	0
NaN	U+1	$\neq 0$

5.3. Operaciones en el sistema de coma flotante. El conjunto de números que se pueden representar y con los que se pueden operar son los números de punto flotante $\mathbb{F}(\beta, t, L, U)$ que es un subconjunto del conjunto de números reales, se debe entonces representar en $\mathbb{F}$ cualquier número con el que se quiera operar, incluso si dos números x e y pertenecen a $\mathbb{F}$ el resultado de una operación entre ellos no necesariamente también. Para cualquier $x \in \mathbb{R}$ se debe elige una representación flotante en $\mathbb{F}$ que se suele indicar por $fl(x)$. Lo habitual es hacer un redondeo de la siguiente forma

$$fl(x) = (-1)^s \cdot (0.a_1 \cdots \hat{a}_t) \cdot \beta^e, \quad \text{con} \quad \tilde{a}_t = \begin{cases} a_t & \text{si} \quad a_{t+1} < \beta/2, \\ a_t + 1 & \text{si} \quad a_{t+1} \geq \beta/2. \end{cases}$$

Esta representación será válida cuando el $L \leq e \leq U$, si el resultado de alguna operación obtiene un exponente menor que L se obtiene un underflow y cuando $e > U$ un overflow que habitualmente interrumpe la ejecución del programa. Finalmente si $x \in \mathbb{F}$ es evidente que $fl(x) = x$.

Los errores de redondeo están acotados porque como se demuestra por ejemplo en el teorema 2.2 de [3]

$$fl(x) = x(1 + \delta) \qquad \text{con} \qquad |\, \delta \,| \leq \frac{1}{2}\beta^{1-t} = \frac{\mathbf{eps}}{2} = \mathbf{u}. \tag{5.3}$$

Esto indica que el error relativo es menor que el valor $\mathbf{u}$ llamado unidad de redondeo; en precisión simple $\mathbf{u} = 2^{-24} \approx 5.96 \times 10^{-8}$ y en la doble $\mathbf{u} = 2^{-53} \approx 1.11 \times 10^{-16}$.

Cuando se opera en punto flotante entre dos números x e y primero se deben redondear los números y después el resultado, por tanto

$$x \odot y = fl(fl(x) \cdot fl(y))$$

donde $\cdot$ ahora indica una suma, resta, multiplicación o división, y $\odot$ denota la correspondiente operación en la aritmética de punto flotante. Si $x, y \in \mathbb{F}$

$$x \odot y = (x \cdot y)(1 + \delta), \quad \text{con} \quad |\delta| < \mathbf{u}.$$

En esta nueva aritmética algunas de las propiedades de la aritmética clásica se conservan, como por ejemplo la conmutatividad, pero otras se pierden, por ejemplo la asociatividad de la suma

$$x \oplus (y \oplus z) \neq (x \oplus y) \oplus z,$$

lo que significa que el orden de sumación es relevante.

Ejemplo 5.1. Tomando las cantidades $x = 1 \times 10^{20}, y = -1 \times 10^{20}$ y $z = 1$ utilizando MATLAB se tiene

$$(x + y) + z = 1.0, \qquad x + (y + z) = 0.0.$$

Para sumar muchos términos en el capítulo 4 de [3] se demuestra que lo mejor es sumar de menor a mayor.

El número elevado de operaciones que se realizan en la mayoría de los algoritmos numéricos, hace que sea muy importante conocer como se propagan los errores de redondeo. En textos clásicos como [2] o [1] se pueden encontrar estudios detallados de la propagación de los errores de redondeo de algunas operaciones elementales como la suma, resta, multiplicación y división. Por otra parte, Wilkinson en [9] introdujo el concepto de análisis regresivo de los errores para estudiar los algoritmos en Álgebra Lineal.

Aunque no es frecuente que los errores de redondeo se compensen unos con otros para dar mejores resultados finales que intermedios, ello es posible. Considere la siguiente función

$$f(x) = \frac{e^x - 1}{x} = \sum_{i=0}^{\infty} \frac{x^i}{(i+1)!}, \tag{5.4}$$

que se evaluará con dos algoritmos diferentes

algoritmo 1		algoritmo 2	
si	$x = 0$	$y = e^x$	
	$f = 1$	si	$y = 1$
si no			$f = 1$
	$f = \frac{e^x - 1}{x}$	si no	
final			$f = \frac{y-1}{\log y}$
		final	

Con MATLAB los resultados son

x	algoritmo1	algoritmo 2
10^{-5}	1.00000500000696	1.00000500001667
10^{-8}	0.99999999392253	1.00000000500000
10^{-10}	1.00000008274037	1.00000000005000
10^{-12}	1.00008890058234	1.00000000000050
10^{-14}	0.99920072216264	1.00000000000001
10^{-15}	1.11022302462516	1.00000000000000
10^{-16}	0	1

En la página 22 de [3] se demuestra que el algoritmo 2 tiene mejores resultados porque hay una cancelación de errores en la división que no ocurre en el otro algoritmo.

Los errores de redondeo pueden incluso beneficiar el comportamiento de algunos algoritmos. El método de la potencia se utiliza para calcular autovectores asociados a autovalores dominantes de matrices, como se trata de un método iterativo, para comenzar debe tomarse un vector arbitrario. Si el vector inicial no tiene componente en la dirección del vector buscado, teóricamente el método diverge, pero los errores de redondeo hacen la componente, aunque pequeña, diferente de cero y esto corrige la divergencia. Para un estudio detallado de la propagación de los errores se recomiendo la sección 1.3 de la referencia [1].

6. Costo operativo y eficiencia

Habitualmente un problema matemático se puede resolver con diferentes métodos numéricos, la pregunta ahora es con cuál de ellos quedarse y con cómo elegir, quizá el método con menor error, o el más rápido, o quizá el más sencillo de programar. Un criterio adecuado es la eficiencia entendida en el sentido de que necesite menos trabaja para que el error cometido sea inferior a una tolerancia dada. La eficiencia depende del tamaño de los errores que luego se llamara orden del método y de su costo operativo que es el número de operaciones necesaria para llegar a la solución. La eficiencia o no de un método numérico es un rasgo muy importante que siempre se debe analizar.

Ejemplo 6.1 (Algoritmo de Horner)**.** Suponga que se debe evaluar para un valor dado z un polinomio de orden cuatro $a_4z^4 + a_3z^3 + a_2z^2 + a_1z + a_0$. El método habitual calcula los sucesivos productos $z^2 = z\cdot z, z^3 = z^2\cdot z, z^4 = z^3\cdot z$ que son tres productos, después calcula los productos por los coeficientes que son otras cuatro,y finalmente las cuatro sumas o restas dependiendo del signo del coeficiente. En total son cuatro sumas o restas y siete multiplicaciones.

Si ahora se opera en otro orden:

$$a_0 + z\cdot(a_1 + z\cdot(a_2 + z\cdot(a_3 + z\cdot a_4))),$$

el resultado es el mismo pero sólo precisa cuatro sumas o restas y cuatro multiplicaciones, se ha reducido de siete a cuatro el número de multiplicaciones.

Para un polinomio de grado N el método clásico requiere N sumas y 2N-1 multiplicaciones mientras que el algoritmo de Holder necesita N sumas y N multiplicaciones, ha reducido a casi mitad el número de productos. Si como es frecuente en un programa se necesita evaluar polinomios en miles de puntos diferentes, usar un método u otro puede suponer mucho tiempo de computación.

7. Normas vectoriales

Definicion 7.1. Una norma vectorial en $\mathbb{C}^n$ es una aplicación $\mathbb{C}^n \mapsto \mathbb{R}$ tal que

a) $||\mathbf{x}|| \geq 0 \quad \forall \mathbf{x} \in \mathbb{C}^n$ y $||\mathbf{x}|| = 0 \Leftrightarrow \mathbf{x} = 0$.
b) $||\alpha\mathbf{x}|| = |\alpha|||\mathbf{x}|| \quad \forall \alpha \in \mathbb{C}, \forall \mathbf{x} \in \mathbb{C}^n$.
c) $||\mathbf{x} + \mathbf{y}|| \leq ||\mathbf{x}|| + ||\mathbf{x}|| \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{C}^n$.

Ejemplo 7.2. La p-norma para $1 \leq p < \infty$, $\quad ||\mathbf{x}||_p = \left(\sum_{j=1}^{n} |x_j|^p \right)^{1/p}$.

Ejemplo 7.3. La norma infinito $\quad ||x||_\infty = \max_{1 \leq j \leq n} |x_j|$.

Teorema 7.4. *Todas las normas en $\mathbb{C}^n$ son equivalentes, es decir, para cada dos normas $||\cdot||_a$ y $||\cdot||_b$ existen constantes $0 < c_1 \leq c_2 < \infty$ tales que*

$$c_1||\mathbf{x}||_a \leq ||\mathbf{x}||_b \leq c_2||x||_a \quad \forall \mathbf{x} \in \mathbb{C}.$$

Definicion 7.5. Un producto interno en $\mathbb{C}^n$ es una aplicación $\mathbb{C}^n \times \mathbb{C}^n \mapsto \mathbb{C}$ tal que

a) $\langle \mathbf{x}, \mathbf{x} \rangle \geq 0 \quad \forall \mathbf{x} \in \mathbb{C}^n$ y $\langle \mathbf{x}, \mathbf{x} \rangle = \mathbf{0} \Leftrightarrow x = 0$.
b) $\langle \mathbf{x}, \mathbf{y} \rangle = \overline{\langle \mathbf{y}, \mathbf{x} \rangle} \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{C}^n$.
c) $\langle \mathbf{x}, \alpha\mathbf{y} \rangle = \alpha \langle \mathbf{x}, \mathbf{y} \rangle \quad \forall \alpha \in \mathbb{C}, \forall \mathbf{x}, \mathbf{y} \in \mathbb{C}^n$.
d) $\langle \mathbf{x}, \mathbf{y} + \mathbf{z} \rangle = \langle \mathbf{x}, \mathbf{y} \rangle + \langle \mathbf{x}, \mathbf{z} \rangle \quad \forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{C}^n$.

Las propiedades c) y d) indican que el producto interno es lineal en la segunda componente y por tanto anti-lineal en la segunda.

Ejemplo 7.6. El producto interior standard en $\mathbb{C}^n$ está dado por

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{j=1}^{n} \overline{x_j} y_j, \qquad \forall \mathbf{x}, \mathbf{y} \in \mathbb{C}^n.$$

Definicion 7.7. Dos vectores x, y son ortogonales si $\langle x, y \rangle = 0$.

Lema 7.8. *Dado un producto interno $\langle \cdot, \cdot \rangle$ en $\mathbb{C}^n$, se define la norma*

$$||x|| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}, \qquad \forall \mathbf{x} \in \mathbb{C}^n$$

Teorema 7.9. *Desigualdad de Cauchy-Schwarz Para todo par de vectores $x, y \in \mathbb{C}^n$*

$$|\langle \mathbf{x}, \mathbf{y} \rangle| \leq ||\mathbf{x}|| \cdot ||\mathbf{y}||.$$

y la igualdad se obtiene cuando x e y son linealmente dependientes.

8. Normas matriciales

Definicion 8.1. Una norma matricial en $\mathbb{C}^{n \times n}$ es una aplicación $\mathbb{C}^{n \times n} \mapsto \mathbb{R}$ tal que

a) $||A|| \geq 0 \quad \forall A \in \mathbb{C}^{n \times n}$ y $||A|| = 0 \Leftrightarrow A = 0$.
b) $||\alpha A|| = |\alpha| \cdot ||A|| \quad \forall \alpha \in \mathbb{C}, \forall A \in \mathbb{C}^{n \times n}$.
c) $||A + B|| \leq ||A|| + ||B|| \quad \forall A, B \in \mathbb{C}^{n \times n}$.
d) $||A \cdot B|| \leq ||A|| \cdot ||B|| \quad \forall A, B \in \mathbb{C}^{n \times n}$.

Definicion 8.2. Dada una norma vectorial $||\cdot||_v$ en $\mathbb{C}^n$ la norma inducida en $\mathbb{C}^{n\times n}$ se define de la forma

$$||A||_m = \max_{x\neq 0} \frac{||A\mathbf{x}||_v}{||\mathbf{x}||_v}.$$

Teorema 8.3. *Para cualquier norma matricial inducida por una norma vectorial*

1. $\rho(A) \leq ||A||_m \quad \forall A \in \mathbb{C}^{n\times n}$
2. $||I||_m = 1.$
3. $||A\mathbf{x}||_v \leq ||A||_m||\mathbf{x}||_v \quad \forall A \in \mathbb{C}^{n\times n}, \mathbf{x} \in \mathbb{C}^n.$

Teorema 8.4. *Para toda matriz $A \in \mathbb{C}^{n\times n}$,*

$$\begin{aligned} ||A||_\infty &= \max_{1\leq i\leq n} \sum_{j=1}^{n} |a_{ij}|. \\ ||A||_1 &= ||A^*||_\infty = \max_{1\leq j\leq n} \sum_{i=1}^{n} |a_{ij}|. \\ ||A||_2^2 &= \rho(A^*A). \end{aligned}$$

Teorema 8.5. *Para cualquier norma matricial $||\cdot||$, $A \in \mathbb{C}^{n\times n}$ y $k \in \mathbb{N}$ se tiene las siguientes desigualdades*

$$\rho(A)^k \leq \rho(A^k) \leq ||A^k|| \leq ||A||^k.$$

Teorema 8.6. *Si la matriz $A \in \mathbb{C}^{n\times n}$ es normal, entonces*

$$\rho(A)^k = ||A^k||_2 = ||A||_2^k \qquad \forall k \in \mathbb{N}.$$

Referencias

1. R. Bulirsch and J. Stoer, *Introduction to Numerical Analysis*, Springer, 1980.
2. G. Forsythe and C.B. Moler, *Computer Solution of Linear Algebraic Systems*, Prentice-Hall, 1967.
3. N.J. Higham, *Accuracy and Stability of Numerical Algoritms*, SIAM, 1996.
4. M.H. Holmes, *Introduction to Scientific Computing and Data Analysis*, Springer, 2016.
5. C.B. Moler, *Numerical Computing with* Matlab, SIAM, 2004.
6. R. Sacco, F. Saleri, and A. Quarteroni, *Numerical Mathematics*, Springer, 2000.
7. J.M. Sanz-Serna, *Diez lecciones de Cálculo Numérico*, Universidad de Valladolid, 2010.
8. L.N. Trefethen and D. Bau, *Numerical Linear Algebra*, SIAM, 1997.
9. J.H. Wilkinson, *Rounding Error in Algebraic Processes*, Prehtice-Hall, 1963.

Dep. Matemáticas de la UPV/EHU
Email address: fernando.vadillo@ehu.es